Intelligent Autonomous Systems Project 1: Color Segmentation

Ethan Glaser

eg492

Color segmentation of an image involves associating pixels with certain classes that represent colors. This strategy can be extended to identify groups of similarly colored pixels that represent objects. In order to identify a specific object in an arbitrary image containing that object, a conditional distribution can be learned for that color or pixel class and then applied to a new pixel (and generalized across an entire image) to identify whether an unknown pixel is within that class. This distribution can be learned using a set of training images, with pixels isolated that represent the class of interest, and a mean and covariance determined for the pixels of that class. Additionally, once an object is identified in an image, the distance away from that camera that the object is can be determined using an inverse relationship between the size (in pixels) of the object in the image and the distance from the camera, related by the focal width of the camera.

This project seeks to identify the distance away from the camera a red barrel is in an image, using the above strategies of pixel segmentation and distance estimation.

This overall task can be broken down into several subcomponents: pixel segmentation and classification, bounding box placement, and distance estimation. The first step in this process was creating a dataset by manually labelling the training set using the RoiPoly Python library with relatively precise classification, which led to a dataset of rgb pixel values that belonged to the red barrel class based on the training images. From this dataset, a single gaussian model could be developed, by first determining the mean RGB pixel value and 3x3 covariate matrix. The single gaussian components could then be applied to a new image in a vectorized format to return a single value for every pixel in the new image. An alternative approach to the single gaussian is the Gaussian Mixture Model, which involves a combination of gaussians that are used to indicate a pixel's status. The implementation of this method was the bulk of the workload for this project, with a function created to take in the number of gaussians, and then once again vectorizing the process to return a single value for every pixel in a test image.

Initially, experimentation was conducted on the original RGB images, which led to somewhat successfully pixel segmentation for both the basic gaussian and gaussian mixture model, but difficulties with very particular scenarios in images – for example a red barrel in a dark room, or outdoor barrel on a gray concrete background. Alternative color spaces were explored, and identified that the HSV color space was effective in these scenarios, but less effective in some of the simpler situations. Therefore, a hybrid model was explored, going through the gaussian mixture model process with both image color spaces and then combining the results and identifying a constant to represent the sum of the different color spaces. An even split between the two color spaces led to the results that were desired, although this ratio is a hyperparameter that could potentially be further tuned.

After the gaussian models were applied to an image, the next step involved post processing on the result to identify clusters of pixels that could represent red barrels. This post-processing was a multi-step process that first involved a binary classification of the result using a single constant (hyperparameter). When the pixel classification was complete, additional processing was done to create bounding boxes, based on the size of a potential barrel, the density of red barrel pixels, and the ratio of

the dimensions of the barrel. This involved 3 additional hyperparameters (minimum barrel size in the image, proportion of box that is red pixels, and box height / width).

There were many hyperparameters involved in this process so far, and considering the effort it took to implement the GMM, a significantly reduced amount of time was put into hyperparameter tuning the model. These hyperparameters included: number of gaussians (experimented on either 2 or 3 and somewhat arbitrarily used 3 for final model), red barrel/not threshold value (settled on 2.5 for hybrid to identify as many red barrel pixels as possible), size of the barrel in pixels (set to $1/450^{th}$ of entire image, and 0.9 as ratio of pixels inside box to consider it (0.9 is high but remember this is after dilation). Each hyperparameter was experimented upon individually, but the final value was selected as a balance between being accurate on the estimated red barrel pixels but missing a lot vs. getting most of the red barrel pixels but also having excess. The post processing logic is definitely holding back the performance and improvements to this will lead to improved performance.

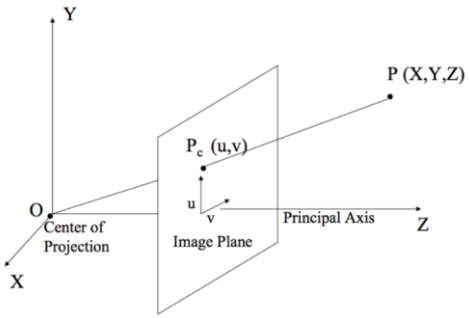
The final step was taking the determined red barrels and identifying the distance from the camera. Figure 1 indicates that there is an inverse relationship between distance from camera and number of pixels of a given dimension of an object, based on the focal length of the image. To estimate the constant that represented this inverse relationship, the training set was once again analyzed, taking the actual distance along with the height and width of the barrel in image to approximate a constant. The average values representing the inverse relationship of these dimensions were 445 for width and 649 for height (445 / pixel width of barrel or 649 / pixel height of barrel were used to approximate the distance from the camera). In some cases these values differed quite a bit, so the maximum was taken to account for parts of the barrel potentially being covered up.

Overall, the finalized pseudocode for the Gaussian Mixture Model implementation was something along the lines of:

1. Read in the red pixel dataset in RGB form
2. Determine the mean, covariance for RGB and convert to HSV and do the same
3. Randomly initialize mean and covariance matrices for specified number of gaussians
4. Read in red pixel barrel training dataset
5. Use current mean and covariance to calculate the g value shown in Figure 2 (vectorized)
6. Determine z from g and use z to update the mean and covariance matrices, see Figure 3 (vectorized)
7. Repeat steps 5-6 until approximate convergence
8. Repeat steps 4-7 on HSV data
9. Read in test image and plug pixel values into gaussian analysis equation in Figure 4 (vectorized)
10. Repeat step 9 for image converted to HSV
11. Normalize the resulting values (result in one value for each pixel)
12. Apply threshold to create binary 2D matrix representing all of the pixels as either red barrel or not
13. Apply dilation to increase continuity of the pixel segmentation
14. Identify potential bounding boxes and keep those that occupy more than $1/450^{th}$ of the entire image, have a red barrel pixel density of 0.9 or greater, and whose height is between 1.2 and 2.1 times the width

15. Calculate the distance from the camera as max(445/pixel width, 659/pixel height)
16. Repeat steps 11-15 for each test image

The algorithm performed well on the training dataset, with a few specific cases where it struggled to identify enough pixels to be dense enough for a bounding box, something covered part of the bounding box, etc. The test set contained some of these outlying images. The performance on the tests set is shown below, with 6 of the 10 images performing as expected. The other 4 failed for various reasons – the first had another red rectangle in the background that was falsely identified to be a barrel, the second was successful in pixel identification but part of the middle of the barrel was cut of, rendering it unable to meet the density cutoff. The other two were rotated and cut off, leading to no bounding box created. Overall the test results suggested that the pixel color segmentation and binary classification were effective, but the post processing method needs further improvement.



$$\frac{u}{f} = \frac{X}{Z} \qquad \frac{v}{f} = \frac{Y}{Z}$$

Figure 1: Size in image compared to actual size, from course slides

$$g_k(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T {\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

Figure 2: GMM g calculation (from lecture slides)

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{z_k} \sum_{i=1}^{N} z_k^i \mathbf{x}_i$$

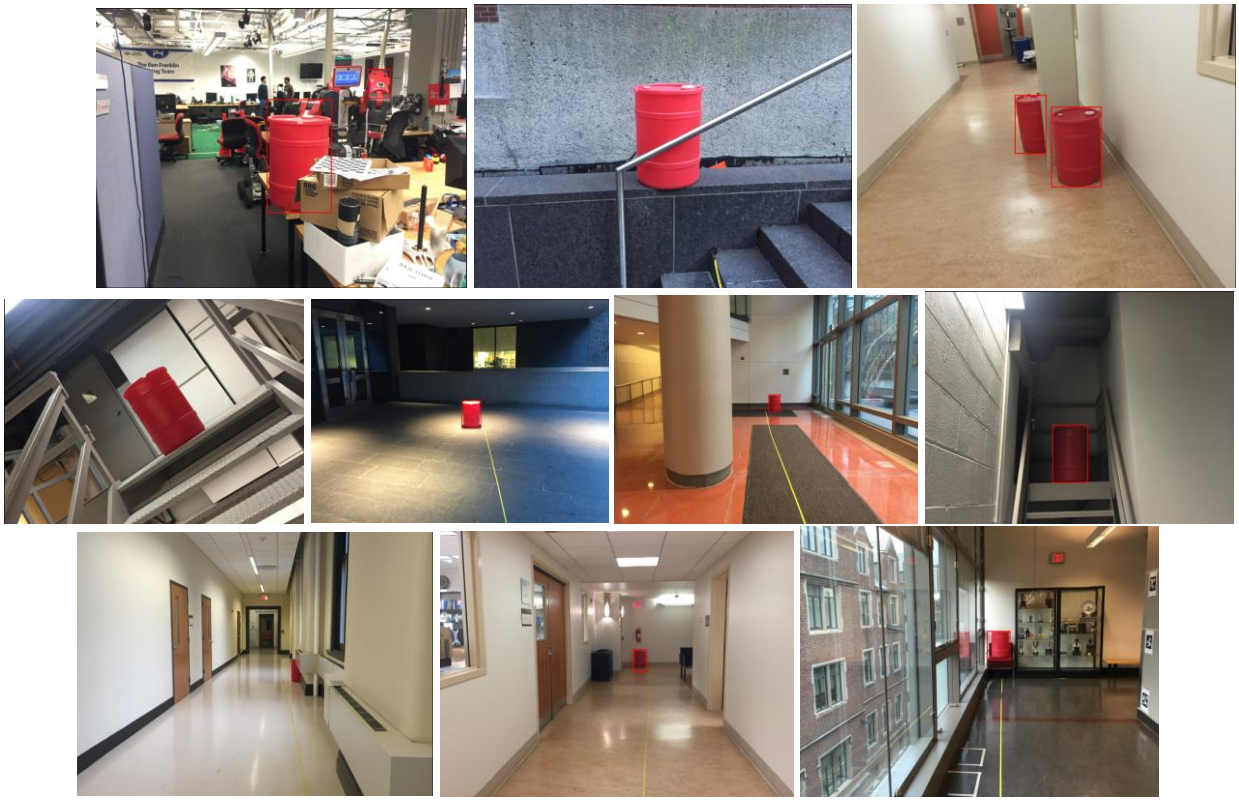
$$\hat{\Sigma}_k = \frac{1}{z_k} \sum_{i=1}^{N} z_k^i (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\intercal$$

$$z_k = \sum_{i=1}^{N} z_k^i$$

Figure 3: GMM mean, covariance, and z calculations (from lecture slides)

$$p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^D \det \Sigma}} \exp\left[-\frac{1}{2}(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})\right]$$

Figure 4: Multivariate Gaussian equation (also from lecture slides)



```
001.png
ImageNo = [0], CentroidX = 917.5, CentroidY = 242.0, Distance = 6.904255319148936
001.png
ImageNo = [0], CentroidX = 659.5, CentroidY = 475.0, Distance = 2.1497584541062804
003.png
ImageNo = [2], CentroidX = 551.0, CentroidY = 380.5, Distance = 4.540816326530612
003.png
003.png
ImageNo = [2], CentroidX = 551.0, CentroidY = 380.5, Distance = 4.540816326530612
003.png
ImageNo = [2], CentroidX = 695.5, CentroidY = 451.5, Distance = 2.8343949044585988
005.png
ImageNo = [4], CentroidX = 645.0, CentroidY = 461.5, Distance = 5.855263157894737
006.png
ImageNo = [5], CentroidX = 632.5, CentroidY = 427.5, Distance = 9.984615384615385
007.png
ImageNo = [6], CentroidX = 565.5, CentroidY = 625.0, Distance = 3.2962962962962963
009.png
ImageNo = [8], CentroidX = 673.0, CentroidY = 433.0, Distance = 10.113636363636363
010.png
ImageNo = [9], CentroidX = 659.5, CentroidY = 399.0, Distance = 6.904255319148936
```

Figure 5: Output images (with bounding box) of test set and corresponding centroids and distances